KARACRIX 入門実用ガイド

3章　KaracrixBuilder によるシステム構築手順

（章別取扱説明書 v3.00）

株式会社 エスアイ創房

■改定履歴

第 3.00 版　2009/11/01

■ おことわり

(1) 本書内容の一部又は全部を、無断で他に転載することは禁止されています。

(2) 本書内容は、将来予告無く変更する場合があります。

目次

3 章　KaracrixBuilder によるシステム構築手順 3-1
3.1　システム設計 3-1
3.2　センサ、アクチュエータの選定と配線 3-1
3.3　ポイントの登録 3-2
3.4　通信制御ドライバ S1 の設定（インストール） 3-4
3.5　監視パネルの作成 3-7
3.6　制御プログラムの作成 3-8
3.7　制御プログラム作成入門 3-9

3章　KaracrixBuilder によるシステム構築手順

KaracrixBuilder でシステム構築をおこなう際の手順を図3.1.1 に示します。

3.1　システム設計

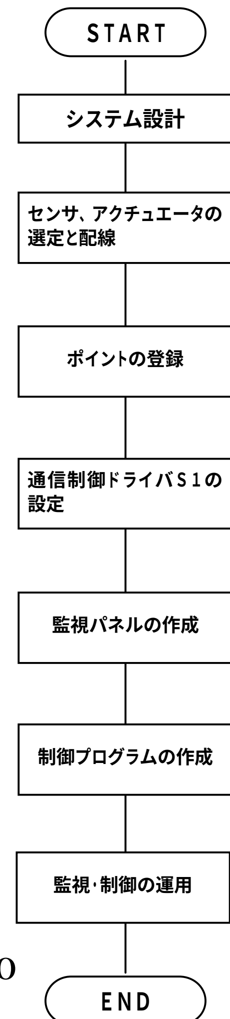


図 3.1.1 システム構築フロー

KaracrixBuilder では、センサやアクチュエータなどの計測・制御対象をどのように構成するのかをシステム設計するところから作業を始めます。計測制御システムの設計とは、センサからの入力情報を収集し目的に応じたデータ処理を施して表示したり、計測データを元にアクチュエータに適当な動作信号を与えるなどの入出力の処理プロセスを整理してプログラム化するまでの一連の作業です。

3.2　センサ、アクチュエータの選定と配線

システム設計で計測制御に使用する情報(温度、照度、水位、電力、接点状態など)の仕様が決定されると、それらの情報を取得するためのセンサと制御機構のアクチュエータの選定を行います。 センサ、アクチュエータの選定は、入手のしやすさと精度の問題と同時に、入出力信号の扱いやすさも選定のポイントとなります。
センサ等をリモート I/O 装置に接続する場合には、装置の電気的入力制限に注意する必要があります。例えばアナログ信号では A/D コンバータの最大入力電圧を超えないよう注意します。アクチュエータの選定では、電力を扱うので動作させる装置に必要な電流容量などを検討します。必要があれば外部緩衝兼駆動装置(リレー) などを別途用意します。
センサ、アクチュエータが用意できたら設置します。センサ、アクチュエータからの配線は、後で接続の確認が容易になるように色分けするなど、整理してリモート I/O 装置に接続します。

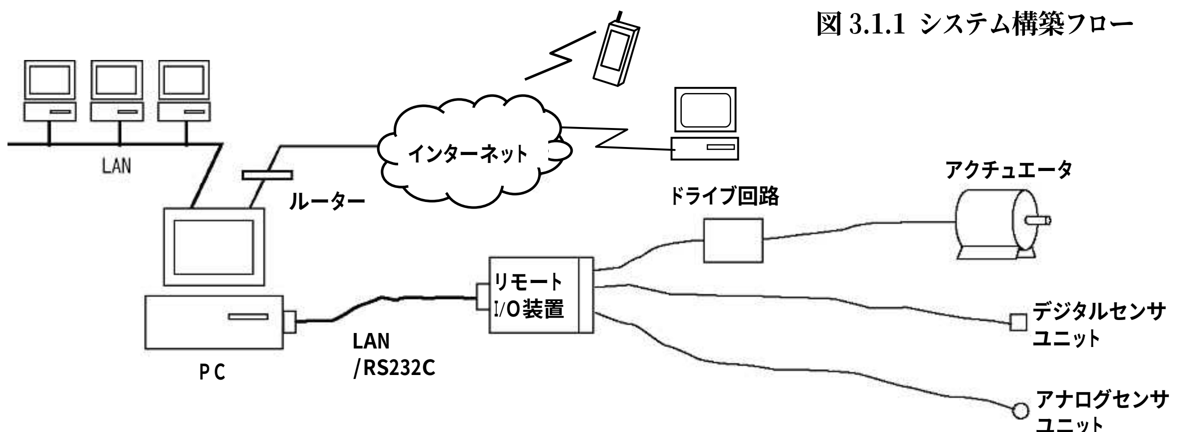


図 3.2.1 センサ、アクチュエータの接続

3.3　ポイントの登録

KaracrixBuilder では、リモート I/O 装置の I/O 端子に接続されたセンサやアクチュエータなどをポイントオブジェクトとして記号化してシステム内部で扱うためポイント登録を行なう必要があります。
KaracrixBuilder の「ポイント登録」画面で、使用するセンサやアクチュエータを種類別にポイントオブジェクトに割り付けます。この作業で、センサから取得した計測データや、アクチュエータに出力する操作データが KaracrixBuilder 内部で管理するオブジェクト ID と対応付けられます。例えば、温度センサを ai001 というオブジェクト ID に対応付けた場合、システム内部では、ai001 と抽象化された ID で温度センサのデータを扱うことができるようになります。*1

KaracrixBuilder-24A を使用している場合には、24 点のポイントオブジェクトを管理することができ、「ポイント登録」画面では 24 点のポイントオブジェクト ID (OBJID) の使用方法がデフォルト (種類固定) で決められています。

デジタル入力 (di001～di006)	接点入力 1～6
デジタル出力 (do001～do004)	リレー (トランジスタ) 出力 1～4
カウンタ入力 (pi001～pi006)	カウンタ入力 1～6
アナログ入力 (ai001～ai004)	アナログ入力 1～4
アナログ出力 (ao001～ao003)	アナログ出力 1～3
イメージ入力 (img001)	イメージ入力 1

また、デジタル入出力やアナログ入出力など扱う入出力の種類別に各種属性情報の設定を適切に行う作業も必要になります。*1
ポイント登録の具体的な設定内容については 4 章以降で解説していますので、ここでは、センサやアクチュエータの入出力をオブジェクトつまりポイントオブジェクト ID (OBJID) として扱う考え方を理解して頂ければ結構です。

*1 設定の詳細は、「KaracrixBuilderV3 システムマニュアル (20 章 オブジェクト環境設定) 」を参照。

以下に「ポイント登録」画面の表示例を示します。

No.	種別	OBJID	タグ名	ポイント名	属性設定	W3	MB
1	DI	di001	T-di001	接点入力1	(ON/OFF) (ALM/-)	*	*
2	DI	di002	T-di002	接点入力2	(ON/OFF) (ALM/-)	*	*
3	DI	di003	T-di003	接点入力3	(ON/OFF) (ALM/-)	*	*
4	DI	di004	T-di004	接点入力4	(ON/OFF) (ALM/-)	*	*
5	DI	di005	T-di005	接点入力5	(ON/OFF) (ALM/-)	*	*
6	DI	di006	T-di006	接点入力6	(ON/OFF) (ALM/-)	*	*
7	DO	do001	T-do001	リレー出力1	(ON/OFF)	*	*
8	DO	do002	T-do002	リレー出力2	(ON/OFF)	*	*
9	DO	do003	T-do003	リレー出力3	(ON/OFF)	*	*
10	DO	do004	T-do004	リレー出力4	(ON/OFF)	*	*
11	PI	pi001	T-pi001	カウンタ入力1	(999999) (10000)	*	*
12	PI	pi002	T-pi002	カウンタ入力2	(999999) (10000)	*	*

図 3.3.1 ポイント登録画面（デジタル入出力（DI,DO））

No.	種別	OBJID	タグ名	ポイント名	属性設定	W3	MB
13	PI	pi003	T-pi003	カウンタ入力3	(999999) (10000)	*	*
14	PI	pi004	T-pi004	カウンタ入力4	(999999) (10000)	*	*
15	PI	pi005	T-pi005	カウンタ入力5	(999999) (10000)	*	*
16	PI	pi006	T-pi006	カウンタ入力6	(999999) (10000)	*	*
17	AI	ai001	T-ai001	アナログ入力1	(100.00/0.00) (-) (%8.2f)	*	*
18	AI	ai002	T-ai002	アナログ入力2	(100.00/0.00) (-) (%8.2f)	*	*
19	AI	ai003	T-ai003	アナログ入力3	(100.00/0.00) (-) (%8.2f)	*	*
20	AI	ai004	T-ai004	アナログ入力4	(100.00/0.00) (-) (%8.2f)	*	*
21	AO	ao001	T-ao001	アナログ出力1	(100.00/0.00) (-) (%8.2f)	*	*
22	AO	ao002	T-ao002	アナログ出力2	(100.00/0.00) (-) (%8.2f)	*	*
23	AO	ao003	T-ao003	アナログ出力3	(100.00/0.00) (-) (%8.2f)	*	*
24	IMG	img001	T-img001	イメージ入力1	(ch=1) (320x240) (jpg=70%)	*	

図 3.3.2 ポイント登録画面（カウンタ入力（PI）、アナログ入出力（AI,AO））

3.4　通信制御ドライバ S1 の設定（インストール）

KaracrixBuilder では、ポイント登録で設定したポイントオブジェクト（対象）の情報は、システム内部でオブジェクトメモリと呼ばれる実体メモリとして参照（読み書き）されます。このオブジェクトメモリに対してユーザープログラムやシステムの監視パネルなどが参照することで計測制御システムが構築されているわけです。
そして、このオブジェクトメモリと外部をつなぐものとして例えば「通信制御ドライバ S1」があります。
通信制御ドライバ S1 は、接続するリモート I/O 装置と直接通信して各種センサからの情報を取得したり、アクチュエータを操作する機能を担います。
通信制御ドライバ S1 のインストールや動作の詳細については 2 章で解説していますので参照して下さい。

通信制御ドライバ S1 は、制御パラメータから接続するリモート I/O 装置の接続情報（IP アドレス、ポート番号）と、I/O 端子とポイントオブジェクト ID（OBJID）のひも付け情報を読み込み動作します。

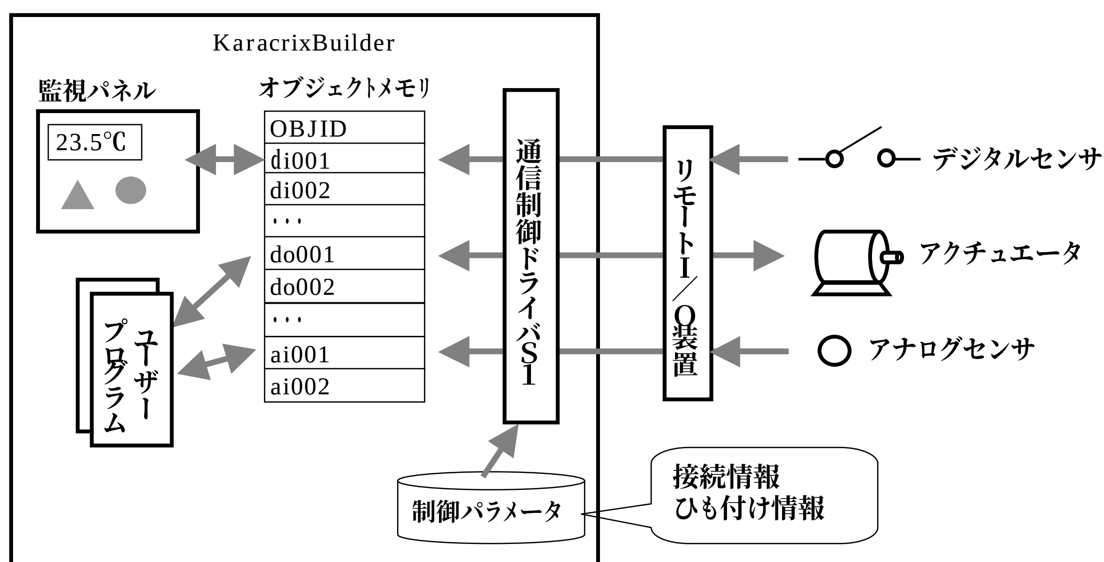


図 3.4.1 オブジェクトメモリを介して連携するユーザープログラムと通信制御ドライバ S1

●通信制御ドライバ S1 の仮実行

ここでは、2章を参照して通信制御ドライバ S1 のインストールとコンパイルが終了しているものとして解説します。リモートI/O 装置は、まだ接続しないで結構ですので試験的に通信制御ドライバ S1 を実行してみましょう。以下の図に示すように、「制御プログラム登録」画面の通信制御ドライバ S1 の"実行"欄を選択して下さい。「プログラムを実行します」と確認ダイアログが表示されますので、"RUN"を選択して下さい。

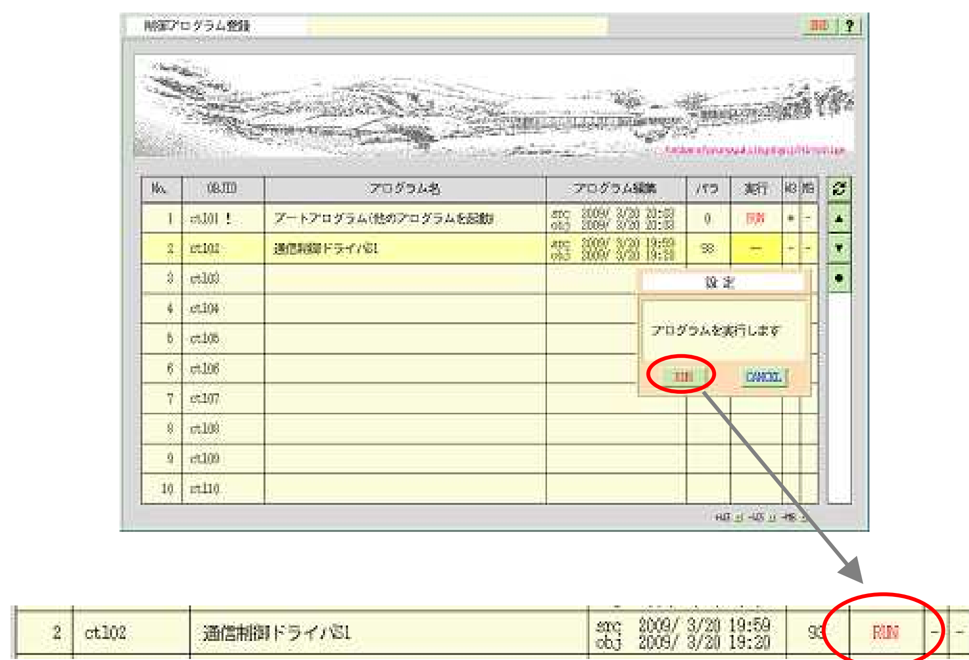


図 3.4.2 通信制御ドライバ S1 を試験的に実行する

"実行"欄に"RUN"と表示されることを確認して下さい。この時点では、リモートI/O 装置は接続していませんので、しばらくすると"RUN"の表示が停止を示す"--"に戻り、プログラムは終了するはずです。

●制御パラメータ編集画面の確認

「通信制御ドライバ　S1」の実行確認ができたら、メインメニューに戻って「制御パラメータ」ボタンから「制御パラメータ入力」画面を表示して下さい。

以下の図に示すように、“通信制御ドライバ　S1”が登録されていると思いますので、その行を選択して「パラメータ編集」画面を表示して見て下さい。

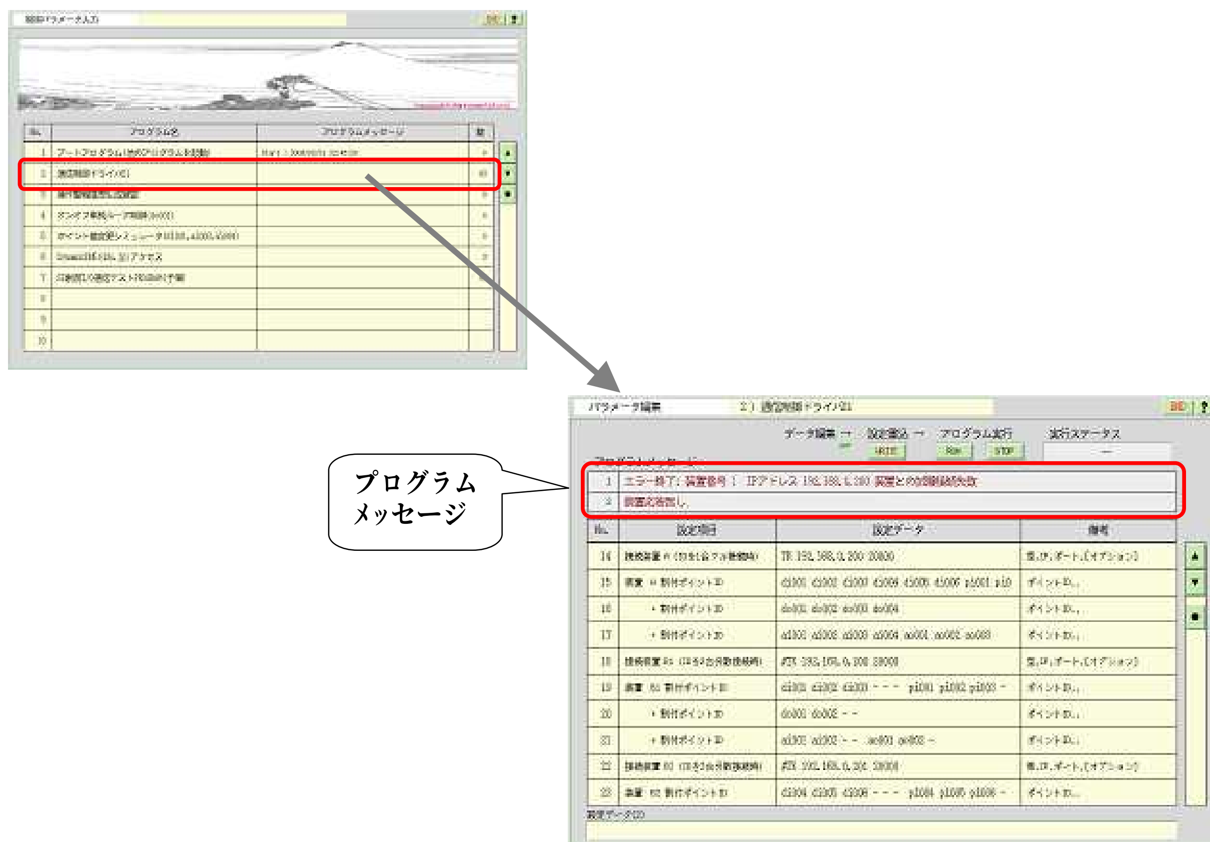


図 3.4.3 パラメータ編集画面の確認

“プログラムメッセージ”表示欄に、以下のようなメッセージが表示されていることを確認して下さい。

1	エラー終了：装置番号 1　IPアドレス 192.168.0.200 装置との試験接続失敗
2	装置応答無し

図 3.4.4 プログラムメッセージの確認

これは、デフォルトの制御パラメータとして設定されている情報を元に、IP アドレス 192.168.0.200 のリモート I/O 装置に接続を試みたが、装置が接続されていないため接続に失敗したことを示しています。4 章以降で、リモート I/O 装置（TK0040A）を実際に接続した時のメッセージについて解説しています。ここでは、確認が目的ですので、これで終了です。

3.5　監視パネルの作成

「監視パネル CAD」を使用して監視パネルを作成します。監視パネルとは、センサやアクチュエータの動作状況をPCのディスプレイ上で確認したり、画面上に表現されたアイコンなどを、マウスでクリックすることによってリモート操作できるしくみを提供するものです。

※監視パネルで作成した画像は、PCや携帯のWebブラウザからも参照できます。

●テスト用監視パネルのインストールについて

弊社のリモートI/O装置をお使いの方は、リモートI/O装置の機能テストで使用するテスト用監視パネルを「監視パネルメニュー」の「監視パネル管理」画面に登録しておく必要があります。
お使いのKaracrixBuilderソフトのバージョンによっては、テスト用監視パネルが既に登録され使える状態になっている場合がありますが、ご利用のリモートI/O装置用のものが登録されていない場合には、弊社HPのダウンロードページにリモートI/O装置別に対応したテスト用監視パネルを用意していますので、これをダウンロードし、KaracrixBuilderユーザーズマニュアル「18章 リソースファイルの入出力」→「18.5 監視パネル&複部品ファイル」の解説に従って「監視パネル管理」画面にインストール(インポート)してください。
なお、インストールする監視パネル名は、本マニュアルの以降の章で共通して引用していますので
“リモートI/Oテスト”としてください。

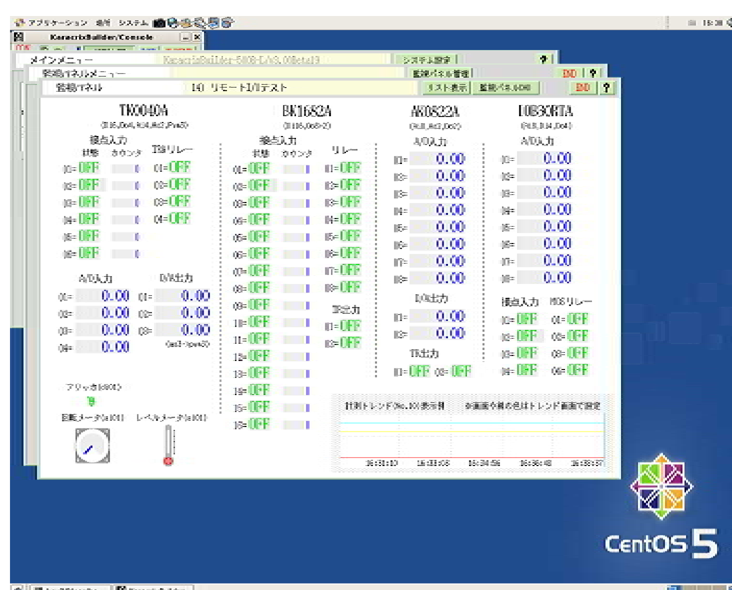


図3.5.1 テスト用監視パネル(リモートI/Oテスト)

3.6 制御プログラムの作成

センサやアクチュエータなど入出力の自動制御を行う場合は、制御プログラムを作成する必要があります。KaracrixBuilder では、センサやアクチュエータをコントロールするプログラムにC言語を使用して作成することができる KCX ライブラリ[*1]を提供しています。各種センサやアクチュエータは、プログラム上では **"ポイントオブジェクト ID"** を使用して扱うことができます。作成したプログラムをコンパイルして実行します。

なお、プログラムの編集は、普段お使いのエディタ(gedit,vi,emacs など)をご利用して作成して頂くのが最も効率的です。外部作成したプログラムは、KaracrixBuilder の制御プログラムとしてインポート(取り込む)して使用することができます[*2]。インポートしたプログラムの小規模な変更や編集には、KaracrixBuilder に付属の簡易エディタ機能をお使い頂くと便利です。

*1 KaracrixBuilderV3 システムマニュアル 「23 章 KCX ライブラリリファレンス」参照
*2 KaracrixBuilderV3 システムマニュアル 「18 章 リソースファイルの入出力」→「18.2 制御プログラムファイル」参照

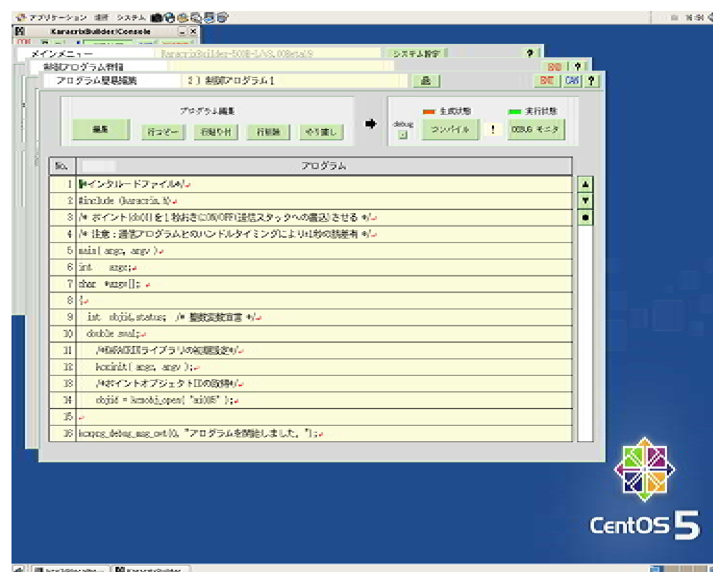


図 3.6.1 制御プログラムの作成

3.7 制御プログラム作成入門

(1)制御プログラムの作成手順

ここでは、KaracrixBuilder 上で制御プログラムを作成する手順の1つを紹介します。
例として温度センサからの入力を使ってある設定温度を超えたら換気ファンのONを行なうという簡単な計測制御システムを考えてみます。先ずは、処理手順を以下に示す様に文章で記述します。

＜処理手順＞
①設定温度を入力する
②温度センサから計測値を入力する
③温度センサの値と設定温度の差を計算する
④差＞0の場合に⑤を実行
⑤換気ファンをONにする

上記の様に行ないたい処理手順を順番に並べて記述することがプログラム作成の第一歩です。次に、処理手順を記述したものを処理フロー図として表現して整理します。処理フロー図を描いてみることで、プログラムの構造が明確になりプログラム作成の見通しが格段によくなります。以下に処理フロー図を示します。

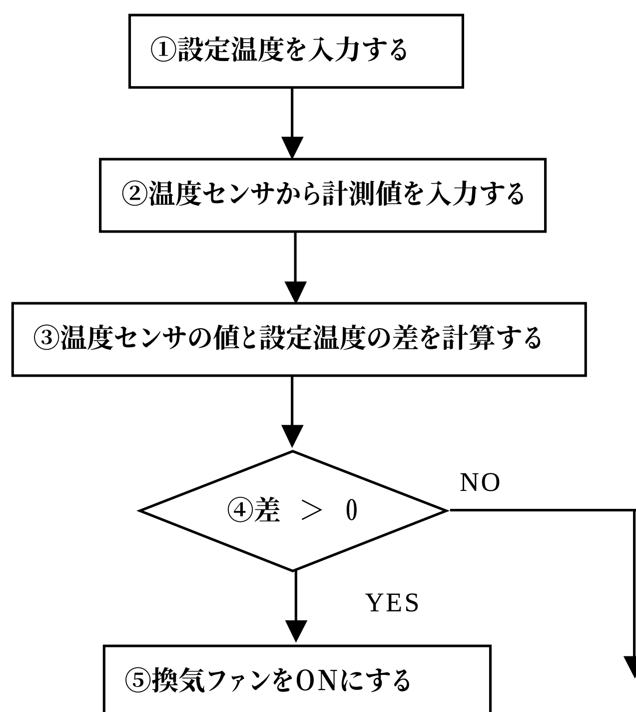


図 3.7.1 温度計測制御システム処理フロー図

⑵ 制御プログラムのプログラミング

前記処理フロー図で表現した手順を、本 KaracrixBuilder 実用ガイド1章の 1.13「ユーザープログラミング簡単入門」の※B の部分に、以下に示すようにプログラムとして記述していきます。

```
#include <karacrix.h>
main( int argc, char *argv[] )
{
        kcxinit( argc, argv );
       ①設定温度を入力する
       ②温度センサから計測値を入力する
       ③温度センサの値と設定温度の差を計算する
       ④差＞0の場合に⑤を実行
       ⑤換気ファンをＯＮにする
}
```

それでは、上記の手順を順番にプログラム記述していきましょう。

①設定温度を入力する

設定温度は、ユーザが指定するパラメータ値ですが、設定温度として例えば“25.5°C”のように実数値（少数含む数値）で指定することにします。プログラムでは、設定した値は後々微調整に利用されるため固定値ではなく変数として扱うことも多いのでここでは、その実数を保存するために double 型の変数を宣言し用います。変数名は、扱う値の意味が判りやすい名前(大小文字交えて長くなっても分かりやすい方が良い)を付けることがコツです。ここでは、この設定温度を越えた場合の判断に使用していますので、Limit_Temp としました。

```
double    Limit_Temp;
```

さらに、変数“Limit_Temp”に値を設定するには、代入演算子（=）を使用して代入文を記述します。例に初期値として“25.5”°Cを設定すると以下の記述になります。C 言語では、文の終わりに“;”（セミコロン）が必要なります。

```
Limit_Temp = 25.5;
```

ここまでの記述を反映したリストは以下のようになります。

```
#include <karacrix.h>
main( int argc, char *argv[] )
{
       double Limit_Temp;      /*  設定温度を格納する実数型変数（宣言文）  */
        kcxinit( argc, argv );   /* KCX ライブラリの初期化「最先頭実行文として必須」（実行文）  */
       Limit_Temp = 25.5;      /*  設定温度の初期値を設定（実行文） */
```

②温度センサから計測値を入力する

ai001 のポイントオブジェクト ID（OBJID）にひも付けられている温度センサの値を取得するときは、kcxobj_open 関数で“ai001”オブジェクトの管理 ID を取得して、その管理 ID を使用し kcxobj_stat_frd 関数を使用して値を取得することができます。ai001 オブジェクトの管理 ID を取得するために、int 型の変数 objid_ai001 を宣言しておきます。

まず、オブジェクト（メモリ）を参照するための管理 ID を kcxobj_open 関数の int 型戻り変数の objid_ai001 に取得します。

次に、kcxobj_stat_frd 関数に objid_ai001 を指定して呼び出すと、fdata 実数型変数に温度センサの値が取得されてきます。

```
int     objid_ai001;
double fdata, diff_temp;
objid_ai001 = kcxobj_open("ai001");
kcxobj_stat_frd( objid_ai001, &fdata );
```

ここまでの記述を反映したリストは以下のようになります。

```
#include <karacrix.h>
main( int argc, char *argv[] )
{
      double Limit_Temp;       /* 設定温度を格納する実数型変数（宣言文） */
      int   objid_ai001;          /* オブジェクト ID を格納する整数型変数（宣言文） */
      double fdata, diff_temp; /* 計測温度と差分を格納する実数型変数（宣言文） */
      kcxinit( argc, argv );     /* KCX ライブラリの初期化「最先頭実行文として必須」（実行文） */
      Limit_Temp = 25.5;       /*設定温度の初期値を設定（実行文） */
      objid_ai001 = kcxobj_open("ai001");    /* ai001 の OBJID を取得（実行文）  */
      kcxobj_stat_frd( objid_ai001, &fdata );   /* ai001 の値を取得する（実行文）  */
```

③温度センサの値と設定温度の差を計算する

計測温度と設定温度の差分を計算します。差分を取得する実数型変数をdouble型でdiff_tempと宣言しています。以下の式で求めます。

```
diff_temp = fdata - Limit_Temp;
```

④差＞0の場合 → ⑤換気ファンをＯＮにする

diff_temp > 0 の条件になった場合に、do001オブジェクトとして登録されている換気ファンにON信号を送ります。ai001と同様に、kcxobj_open関数で“do001”オブジェクトの管理IDを取得します。do001オブジェクトの管理IDを取得するために、int型の変数objid_do001を宣言しています。
その管理IDを用いてkcxobj_sndistat_tokcx関数を使用してコマンドキューに操作データを登録します。操作データに、“1”を設定すると換気ファンがON(リレー閉(ショート))するようになっています。あとは、協調動作している「通信制御ドライバS1」がリモートI/O装置に操作データを送信して換気ファンをONにしてくれます。以下の記述を追加します。

```
int    objid_do001;
  objid_do001 = kcx_open(“do001”);    /* do001 の OBJID を取得 */
  if( diff_temp > 0 ){
        kcxobj_sndistat_tokcx( objid_do001, 1 );
  }
```

ここまでの記述を反映したリストは以下のようになります。

```
#include <karacrix.h>
main( int argc, char *argv[]   )
{
       double Limit_Temp;     /* 設定温度を格納する実数型変数(宣言文) */
       int        objid_ai001;       /* オブジェクトIDを格納する整数型変数(宣言文) */
       int        objid_do001;     /* オブジェクトIDを格納する整数型変数(宣言文) */
       double fdata, diff_temp; /* 計測温度と差分を格納する実数型変数(宣言文) */
        kcxinit( argc, argv );    /* KCXライブラリの初期化「先頭実行文として必須」(実行文) */
       Limit_Temp = 25.5;     /* 設定温度の初期値を設定(実行文)  */
       objid_ai001 = kcxobj_open(“ai001”);     /* ai001のOBJIDを取得(実行文)   */
       objid_do001 = kcxobj_open(“do001”);    /* do001のOBJIDを取得(実行文)   */
        kcxobj_stat_frd( objid_ai001, &fdata );    /* ai001の値を取得する(実行文)   */
         diff_temp = fdata - Limit_Temp;          /* 設定温度と計測値の差分を計算(実行文) */
       if( diff_temp > 0 ){                        /* 計測値が設定値を超えている場合(実行文) */
             kcxobj_sndistat_tokcx( objid_do001, 1 );    /* コマンドキューに操作値を登録(実行文) */
       }
 }
```

以上、簡単な例でKaracrixBuilderの制御プログラムの作成手順を紹介しましたが、KCXライブラリの使い方などの雰囲気をつかんで頂けましたでしょうか。